Authenticated Booting, Remote Attestation, Sealed Memory aka “Trusted Computing”

Hermann Härtig
Technische Universität Dresden
Summer Semester 2007
Goals

Understand:
• authenticated booting
• the difference to (closed) secure booting
• remote attestation
• sealed memory

Learn
• to find out about TCPA/TCG documents TPMs etc
Some terms

Secure Booting
Authenticated Booting
(Remote) Attestation
Sealed Memory
Late launch / dynamic root of trust
Trusted Computing
Trusted Computing Base

Attention:
terminology has changed ...
Trusted Computing (Base)

Trusted Computing Base
The set of all components, hardware, software, procedures, that must be relied upon to enforce a security policy

Trusted Computing (TC)
A particular technology comprised of authenticated booting, remote attestation and sealed memory
TC key problems

• Can running certain SW be prevented?
• Which computer system do I communicate with?
• Which stack of Software is running?
  • in front of me?
  • on my server somewhere?
• Can I restrict access to certain secrets (keys) to certain programs?
Trusted Computing Terminology

Measuring

“process of obtaining metrics of platform characteristics”

Examples Hash- Codes of SW

Attestation

“vouching for accuracy of information”

Sealed Memory

binding information to a configuration
DRM: Trust ./ No Trust in end user

{Digital Content}K

Decoder → TV

K

Internet
An Example Application: DRM

- „Digital Content“ is encrypted using symmetric key
- Smart-Card
  - contains key
  - authenticates device
  - delivers key only after successful authentication
- Assumptions
  - Smart Card can protect the key
  - „allowed“ OS can protect the key
  - OS cannot be exchanged
Notation

$SK^\text{priv}$ $Sk^\text{pub}$ Asymmetric key pair of some entity $S$

$\{ M \}Sk^\text{priv}$ Digital Signature for message $M$ using the private key of signer $S$

$H(M)$ Collision-Resistant Hash

Certificate by authority $Ca$:

$\{ \text{ID, } SK^\text{pub}, \text{ other properties } \} \ CaK^\text{priv}$
Identification of Software

Program vendor: Foosoft FS

- $H(\text{Program})$
- $\{\text{Program}, \text{ID- Program}\}^{FSK_{\text{priv}}}$

use $FSK^{\text{pub}}$ to check
Tamper-resistant black box (TRB)

- CPU
- Memory

Non-Volatile Memory:

Platform Configuration Registers:

Volatile memory:
Ways to “burn in” the OS or secure booting

• Read- Only Memory
• Allowed H(OS) in NV memory preset by manufacturer
  • load OS- Code
  • compare H(loaded OS code) to preset H(OS)
  • abort if different
• Preset FSK_{pub} in NV memory preset by manufacturer
  • load OS- Code
  • check signature of loaded OS-Code using FSK_{pub}
  • abort if check fails
Authenticated Booting (AB)

Phases:

• Preparation by Manufacturers (TRB and OS)
• Booting & “Measuring”
• Remote attestation
Authenticated Booting (AB)

<table>
<thead>
<tr>
<th>CPU</th>
<th>Non-Volatile Memory:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Endorsement Key” EK</td>
</tr>
<tr>
<td></td>
<td>preset by Manufacturer</td>
</tr>
</tbody>
</table>

Platform Configuration Registers:
Hash-Code obtained during boot

Volatile memory:
Vendors of TRB and OS

TRB generates key pair: „Endorsement Key“ (EK)
stores in TRB NV Memory: $E_K^{priv}$
emits: $E_K^{pub}$

TRB vendor certifies: \{“a valid EK“, $E_K^{pub}$\}$TVK^{priv}$

OS-Vendor certifies: \{“a valid OS“, $H(OS)$\}$OSVK^{priv}$

serve as identifiers: $E_K^{pub}$ and $H(OS)$
Booting & Attestation

**Booting:**
TRB “measures” OS- Code (computes $H(\text{OS-Code})$)
stores in PCR
no other way to write PCR

**Attestation:**
Challenge: nonce
TRB generates Response:

$\{\text{PCR, nonce}\}E_{K^{\text{priv}}}$
Remaining problems

Now we know identities: $H(\text{loaded-OS})$ and $EK^{\text{pub}}$

Problems to solve:

• OS versioning
• Remote attestation on each message (what about reboot?)
• not only OS on platform (SW stacks or trees)
• Privacy: remote attestation always reveals $EK^{\text{pub}}$
• Black box to big
• Sealed memory
AB (Variant 2, allow OS versions)

<table>
<thead>
<tr>
<th>CPU</th>
<th>Non-Volatile Memory: “Endorsement Key” EK preset by Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>Platform Configuration Registers: OSK&lt;sup&gt;pub&lt;/sup&gt; used to check OS</td>
</tr>
<tr>
<td></td>
<td>Volatile memory:</td>
</tr>
</tbody>
</table>
Vendors of TRB and OS

TRB generates key pair:
- stores in TRB NV Memory: \( EK_{\text{priv}} \)
- emits: \( EK_{\text{pub}} \)

TRB vendor certifies:
{“a valid EK”, \( EK_{\text{pub}} \) TVK_{\text{priv}}}

OS-Vendor certifies:
{“a valid OS”, \( OSK_{\text{pub}} \) OSVK_{\text{priv}}}

and digns OS-Code:
\{OS-Code\} OSK_{\text{priv}}

serve as identifiers:
\( EK_{\text{pub}} \) and \( OSK_{\text{pub}} \)
Booting & Attestation (Variant 2)

Booting:
TRB checks OS-Code using some $\text{OSK}^{\text{pub}}$
stores $\text{OSK}^{\text{pub}}$ in PCR
no other way to write PCR

Attestation:
Challenge: nonce
TRB generates Response:

$\{\text{PCR, nonce}\}^{\text{EK}}_{\text{priv}}$
AB (Variant 3, check for reboot)

Motivation:

\[ \{ \text{OSK}^{\text{pub}}, \text{nonce} \} \text{EK}^{\text{priv}} \]
\[ \{ \text{H(OS)}, \text{nonce} \} \text{EK}^{\text{priv}} \]

always requires access to and usage of EK

Instead:

create new keypair on every reboot:

\[ \text{OSrunningK}^{\text{priv}} \quad \text{OSrunningK}^{\text{pub}} \]
Booting (Variant 3)

Booting:

TRB checks OS-Code using some $OSK_{pub}$

stores $OSK_{pub}$ in PCR

creates $OS_{running}K_{keypair}$

certifies: \( \{ OS_{running}K_{pub}, H(OS) \} EK_{priv} \)
Attestation (Variant 3)

Attestation:

Challenge: nonce

OS generates response:

\{ OSrunningK^{pub}, H(OS)\}EK^{priv}

\{nonce\} OsrunningK^{priv}
Attestation:

Challenge: nonce

OS generates response:

\{ OSrunningK^{\text{pub}}, H(\text{OS}) \}^{\text{EK}^{\text{priv}}}

\{\text{nonce} \} \text{ OsrunningK}^{\text{priv}}

use OSrunningK keypair to establish secure channel
Assumptions

TRB can protect: EK, PCR

OS can protect: OSrunningK_{priv}

Rebooting destroys content of

• PCR and Memory Holding OSrunningK_{priv}
Software stacks and trees

- Application
  - GUI
    - OS Code
      - OS Loader
        - ROOT

- Application
  - GUI
    - OS Code
      - OS Loader
        - ROOT

- Application
  - GUI
    - OS Code
      - OS Loader
        - ROOT
Software stacks and trees

“Extend” Operation

- stack: $\text{PCR}_n = H(\text{PCR}_{n-1} \ | \ | \ \text{value})$
- tree: difficult (unpublished ?)

Key pairs:
- OS controls applications -> generate key pair per application
- OS certifies
  - $\{ \text{Application 1}, \text{App1K}_{\text{pub}} \} \ \text{OSrunningK}_{\text{priv}}$
  - $\{ \text{Application 2}, \text{App2K}_{\text{pub}} \} \ \text{OSrunningK}_{\text{priv}}$
Remote Attestation and Privacy

Remote attestation reveals platform identity: $E^E_{K_{pub}}$

add intermediate step:

- Attestation Identity Key (AIK)
- Trusted third party as anonymizer (TTP)
Remote Attestation and Privacy

CPU

Memory

Non-Volatile Memory:
EK preset by Manufacturer
AIK signed by third party

Platform Configuration Registers:

Volatile memory:
Remote Attestation and Privacy

Generate AIK in TRB

send \{ AIK \} EK_{\text{priv}} to trusted third party

third party certifies: \{ AIK, "good ID" \} TTPK_{\text{priv}}

AIK used instead of EK during remote attestation,
response:
\{ AIK, "good ID" \} TTPK_{\text{priv}}
\{ OSrunningK_{\text{pub}}, H(OS) \} AIK_{\text{priv}}
\{ nonce \} OSrunningK_{\text{priv}}
Late Launch

Use arbitrary SW to start system and load all SW

provide specific instruction to enter “secure mode”
- set HW in specific state (stop all processors, IO, ...)
- Measure “root of trust” SW
- store measurement in PCR

AMD: “skinit” (Hash) arbitrary root of trust
Intel: “senter” (must be signed by chip set manufacturer)
Sealed Memory

Bind sensitive information to specific configuration
(for example: keys to specific machine, specific OS)

Provide information using secure channels

How to store information in the absence of communication channels?
Tamperresistant black box (TRB)

- **CPU**
- **Memory**

**Non-Volatile Memory:**
- storage key

**Platform Configuration Registers:**
- Hash(OS)

**Volatile memory:**
Sealed Memory

Tamperresistant black box

PCR: H(OS)

- Microsoft
- SUSE
- MyOwn

add/delete entry
read
write
Sealed Memory

Seal(SW config, message):
• encrypt("SW config, message", Storage-Key)

Unseal(sealed message):
• decrypt("sealed message", Storage-Key) -> "SW config, message"
• If SW config == PCR then emit message else abort
Migration?

How to transfer information from one TRB to another for example: key for decryption of videos

- Send information to third party
- Destroy information locally and prove to third party
- Thirds party provides information to another entity
Tamper Resistant Box?

IBM 4758 ...

“Trusted Platform Modules”
TPM

- NV Store
- PCR
- EK
- AIK
- Internal Program
- IO

- SHA-1
- RSA
- Key gen
- Random number gen
References

Specifications:

https://www.trustedcomputinggroup.org/
groups/TCG_1_3_Architecture_Overview.pdf

Important Foundational Paper:

Authentication in distributed systems: theory and practice
Butler Lampson, Martin Abadi, Michael Burrows, Edward Wobber
ACM Transactions on Computer Systems (TOCS)